PadhAI: 6 Jars of Sigmoid Neuron

One Fourth Labs

Dealing with more than 2 parameters

What happens when we have more than 2 parameters

1. Consider the following dataset

ER_visits	Narcotics	Pain	TotalVisits
0	2	6	11
1	1	4	25
0	0	5	10
1	3	5	7

2. Then,
   a. $z = \Sigma^{n}_{i=1} w_i x_i$
   b. Or z = (w_1 * ER_visits) + (w_2 * Narcotics) + (w_3 * Pain) + (w_4 * TotalVisits) + b
   c. $\hat{y} = \frac{1}{1+e^{-z}}$
3. So the algorithm is as follows
   a. **Initialise:** w_1, w_2, w_3, w_4 and b randomly
   b. **Iterate over data**
      i. Compute $\hat{y}$
      ii. Compute L(w,b)
      iii. $w_1 = w_1 - \eta \Delta w_1$
      iv. $w_2 = w_2 - \eta \Delta w_2$
      v. $w_3 = w_3 - \eta \Delta w_3$
      vi. $w_4 = w_4 - \eta \Delta w_4$
      vii. $b = b + \eta \Delta b$
      viii. Where $\Delta w_j = \Sigma^{m}_{i=1} (\hat{y} - y) * (\hat{y}) * (1 - \hat{y}) * x_{ij}$
   c. **Till satisfied**
      i. Number of epochs is reached (ie 1000 passes/epochs)
      ii. Continue till Loss < ε (some defined value)
      iii. Continue till $Loss(w,b)_{t+1} \approx Loss(w,b)_t$
   d. A few of the functions from the code also change, namely

```
def f(w, b, x):
    #Sigmoid with parameters w and b
    #Here we do a dot product between vector w and x
    return 1.0 / (1.0 + np.exp(-(np.dot(w, x) + b)))

def grad_w_i(w, b, x, y, i):
    #Here we add i to denote the i-th feature of
    fx = f(w, b, x)
    return (fx - y) * fx * (1 - fx) * x[i]
```

   e. The function do_grad_descent also changes, but we will figure it out in the practical implementation